A Self-Adaptive Recombination Method in Evolutionary Algorithms for Solving Epistatic Problems

Natalia Stark, Carolina Salto

Facultad de Ingeniería, Universidad Nacional de La Pampa
Calle 110 esq. 9, General Pico (6360)
La Pampa, Argentina.
e-mail: {nstark,saltoc}@ing.unlpam.edu.ar

Abstract. There are many different forms of recombination operators available in literature. However, it is difficult to determine a priori which one is the best suited for a given problem. This issue encourages us to propose an adaptive evolutionary algorithm to solve the NK landscape problem, which dynamically selects the recombination operator from an operator pool during the evolution; this removes the need of specifying a single recombinator operator ad-hoc. We compare the performance of our adaptive proposal against traditional evolutionary algorithms in a numerical way. Our experiments show that the simple adaptive mechanism has a good performance among all the evaluated ones on high dimensional landscapes with an additional reduction in pretuning time.

I. Introduction

One of the major issues in the design and implementation of robust genetic algorithms (GA) to solve a problem is the selection of the appropriate recombination operator, in order to obtain good quality of solutions. To further complicate the issue, many different forms of recombination exist. Traditionally, GAs have relied upon one- and two-point crossover operators [11]. But there are many situations in which having a higher number of crossover points is beneficial. Perhaps the most surprising result is the effectiveness of uniform crossover, an operator that produces on the average $L/2$ crossings on strings of length L [24,7].

Currently there are not techniques for selecting which operators to use before the GA is initiated. There are at least two possible approaches to this problem. The first is to make an extensive experimentation to determine, in an empirical way, which operator actually gives the best results for the problem at hand; this task implies an extra set up time, which is computationally expensive. The second approach is to have an adaptive mechanism in which the EA selects the operators it will be use. This paper concentrates on *self-adaptive* approaches, in which the EA itself selects the recombination operators. Self-adaptation is not without precedent, many approaches have been proposed for adjusting GA parameters such as mutation probability [5], crossover probability [3,10], crossover operator [21,26], and population size [8] in the course of a genetic algorithm run.

The purposes of this paper are to propose an adaptive mechanism for controlling the use of crossover in an EA, while the EA is simultaneously solving a problem, and to investigate how this approach can influence the performance of the EAs. The new adaptive mechanism chooses automatically between two different forms of recombination (two-point and uniform crossover) by using a simple and inexpensive criterion based on traditional statistics of the search process. Mutation is assumed to be used at some set rate and is not adapted by the EA. This approach not only allows for the simultaneous exploration of both the problem space and some space of different EAs, but also it tends to alleviate the time required to tune the algorithm parameters.

In order to evaluate the performance of our proposed algorithm, we will use the NK model [14], which is a useful general model both for investigating the structural properties of landscapes and for evaluating the performance of evolutionary algorithms. The NK model is based upon the idea of genetic linkage. Each gene in a genotype makes a contribution towards the fitness contribution of the genotype as a whole, however, the fitness contribution of each gene may be dependent not only on its own allele, but also on the alleles of any number of other genes. As the number of these epistatic connections between genes increases the resulting fitness landscape is changed from being relatively smooth and predictable to increasingly rugged and random. The NK model has been used in Biology and Physics [15,16,22,25], in business environments to model the evolution of organizations [13,17], among others. In the evolutionary computation fields, the NK model has been used as a benchmark for evaluating various encoding schemes and genetic operators on evolutionary algorithms [1,6,12,19]. Thus, it is clear from all this that NK is a problem worth of study.

The outline of the paper is as follows. Section II focuses in the motivations that lead us to propose this adaptive approach. Section III presents the outline of the current implementation of our adaptive algorithm. In Section IV presents a description of problem used in this work to evaluate the performance of our adaptive algorithm. In Section V, we present the parameterization used here. In the next section, we analyze the results from a numerical point of view. Finally, we summarize the conclusions and discuss several lines for future research in Section VII.

II. Motivation

The present availability of genetic operators for binary representations allows the possibility of using different degrees of exploration and exploitation in the same algorithm.

Analyzing the pool of recombination operators available from the literature for binary representations, there is no clear winner [9]. Each of these recombination operators is particularly useful for some classes of problems and poor for other problems. This strengthens our idea of using self-adaptive algorithms, where the algorithm determines the more adequate operator to apply depending of the state of the evolutionary process.

The one-point crossover and the uniform crossover are the selected recombination operators to be used in this work, because they are the basic recombination operators applied currently more often in present works. One-point crossover (1X) selects one crossing point and left segments are exchange, while other segment remains unchanged. Uniform crossover (UX) exchanges genes rather than segments; it can combine features regardless their relative location.

One-point crossover is the least disruptive of material, while uniform crossover is the most disruptive. Also, Booker [4] has noted that, in terms of positional and distributional bias, both one-point and uniform crossover are considerably different. Thus, it is natural to allow the GA to explore a relative mixture of these two operators, the motivation being that different mixtures will represent different intermediate search characteristics between the two extremes.

In literature, many researchers have proposed various adaptation techniques to crossover designed to enhance GA's capabilities [8]. Schaffer et *al.* [18] used a self-adaptive approach where the points at which crossover is allowed to cut and splice material, is adapted. The mechanism appends an additional L bits to each individuals, which is used to determine crossover points at each locus. Spears [21] considered the use of an extra bit to the end of every individual to self-adapt crossover, but instead of searching the large space of n-point crossover distributions, it considers only two forms of crossover, two-point and uniform. Yang [26] proposed a statistics-based adaptive non-uniform crossover (SANUX), which uses the statistics information of the alleles in each locus to adaptively calculate the swapping probability of that locus for crossover operation.

Other works involve the adaptation of the crossover probability instead of adjusting the selection of the operator to be used. In Srinivas et *al.* [23], the proposed adaptive GA varies the crossover probability depending on the fitness values of the solutions. Each chromosome has its own probability of crossover and mutation.

III. Our Proposed Adaptive Genetic Algorithm

In this section we present the characteristics of the adaptive genetic algorithm devised, which dynamically adjust the recombination operator during the evolutionary process (see the structure of the proposed algorithm in Algorithm 1). The algorithm creates an initial population P of μ solutions in a random way, and then evaluates these solutions. After that, the population goes into a cycle where it undertakes evolution, which means the application of genetic operators, to create λ offspring. Finally, each iteration ends by selecting μ individuals to build up the new population from the set of $(\mu+\lambda)$ existing ones. The best solution is identified as the best individual ever found which maximizes the fitness function.

The new adaptive proposal consists in the dynamic selection of the recombination operator that will be applied, taking as criterion the convergence speed. A major advantage of this adaptive criteria lies in that it is not necessary to set it to any ad-hoc value, because a dynamical setting of which is the most appropriate one is done during the search. Also, we focus in reducing the overhead to a minimum, so as to finally allow savings in the numerical effort which can be used later to solve the

Algorithm 1 Basic GA

```
t = 0; {current evaluation}
initialize(P(t));
evaluate(P(t));
  while{(t < max_generations) do
    P'(t) = recombinate(P(t));
    P'(t) = mutate(P'(t));
    evaluate (P'(t));
    P(t+1) = select new population from P'(t) ∪ P´(t);
    t = t + 1;
  end while
```

actual optimization problem.

The execution is initialized by randomly choosing a recombination operator. The criterion to change the recombination operator, used each generation is a function of the average fitness of the population, which is a possible measure of the convergence speed. Following the ideas presented in [2], we define Δf as the difference between the average fitness values in generation t and t - 1: $\Delta f = \overline{f_t} - \overline{f_{t-1}}$. We calculate the average of this difference during k consecutive generations (k=10); the average is denoted as $\overline{\Delta f} = \sum_{t=1}^{10} \Delta f$. If the difference $\overline{\Delta f}$ decreases at least by a factor of ε: $\overline{\Delta f}_k - \overline{\Delta f}_{k-1} \leq \varepsilon \overline{\Delta f}_{k-1}$ (threshold value $\varepsilon \in [0,1]$), it means that the algorithm evolves slowly and the local exploitation will be increased, hence the algorithm have to apply a less disruptive recombination operator as the one-point crossover (1X). On the contrary, if that difference increases by a factor greater than (1-ε) : $\overline{\Delta f}_k - \overline{\Delta f}_{k-1} > (1 - \varepsilon) \overline{\Delta f}_{k-1}$, then the search is proceeding too fast and there exists the possibility of a quickly lost of diversity, hence the operator to be applied will be the uniform crossover (UX) in order to promote exploration in next generations. Algorithm 2 describes the basic adaptive pattern. This criterion is inexpensive to measure, since it checks simple conditions based on information already available in any standard GA, like the mean fitness.

Algorithm 2 Pattern for our dynamic adaptive criteria

```
if  $\overline{\Delta f}_k < (1+\varepsilon)\, \overline{\Delta f}_{k-1}$ then
     P'(t) = 1X(P(t));
else
     P'(t) = UX(P(t));
end if
```

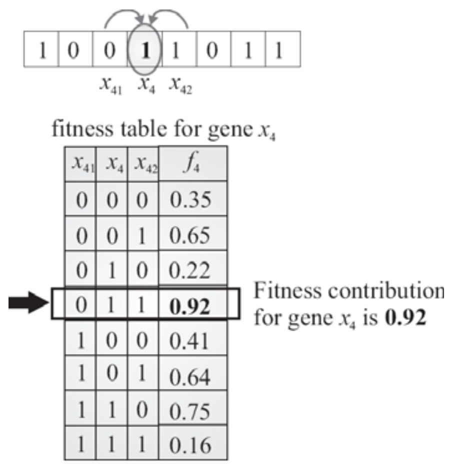
IV. NK-Landscapes

An NK-landscape is a fitness function f: $\{ 0,1 \}^N \rightarrow R$ on binary strings, where N is the bit string length and K is the number of bits in the string that epistatically interact with each bit, i.e., K stands for the number of other genes that epistatically affect the contribution of each gene to the overall fitness value of the string. Each gene x_i, where $1 = x_i = N$, contributes to the total fitness of the genotype depending on the value of its allele and on those of each of the *K* other genes to which it is linked. Thus *K* must fall between 0 and *N*-1. For *K* = 0, there are no interaction among genes and a single-peak landscape is obtained; in the other extreme (for *K*= *N*-1), all genes interact each other in constructing the fitness landscape, so a completely random landscape is obtained (a maximally rugged landscape). Varying K from 0 to *N*-1 gives a family of increasingly rugged multi-peaked landscapes.

The fitness value for the entire genotype is given as the average of the fitness contribution of each locus f_i by:

$$f(x) = \frac{1}{N} \sum\nolimits_{i=1}^{N} f_i(x_i, x_{i_1}, \ldots x_{i_K}) \qquad (1)$$

where $\{x_{i_1}, \ldots x_{i_K}\}$ $\{x_1, \ldots x_{i-1}, x_{i+1}, \ldots, N\}$ are the *K* genes interacting with gene x_i in the genotype *x*. The other *K* epistatic genes could be chosen in any number of ways from the *N* genes in the genotype. Kauffman [15] investigated two possibilities: *adjacent neighborhoods*, where *K* genes nearest to gene x_i on the chromosome are chosen, particularly a gene interacts with *K*/2 left and *K*/2 right adjacent genes; and *random neighborhoods*, where these *K* other genes are chosen randomly on the chromosome. In this work we adopted the first type of neighbourhood and considered circular genotypes to avoid boundary effects. The fitness contribution f_i of x_i is taken at random from a uniform distribution [0.0, 1.0] and depends upon its value and those present in *K* other genes among the *N*. Each gene has associated a *fitness table*, mapping each of the 2^{K+1} possible combinations of alleles to a random, real value number in the range [0,1]. Figure 1 gives an example of the fitness function f_4 (x_4, x_{4_1}, x_{4_2}) associated to gene x_4 for *N*=8 and *K*=2. Gene x_4 is linked to two other genes, in this case, the gene to both sides, x_{4_1} and x_{4_2}.


1 0 0 1 1 0 1 1
x_{41} x_4 x_{42}
fitness table for gene x_4
Fitness contribution for gene x_4 is **0.92**

x_{41}	x_4	x_{42}	f_4
0	0	0	0.35
0	0	1	0.65
0	1	0	0.22
0	1	1	**0.92**
1	0	0	0.41
1	0	1	0.64
1	1	0	0.75
1	1	1	0.16

Figure 1. An example of a genotype and a fitness table associated to gene x_4 for a problem with *N*=8 and *K*=2.

V. Experimental Setup

In order to observe and compare the effect on performance of the self-adaptive proposal (GA_Adap), we use the following traditional GAs: (i) a GA running with UX as recombination operator in traditional fashion, denoted as (GA_UX); (ii) a GA with one-point crossover, denoted as GA_1X.

In order to perform subsequent comparisons, the whole population of all evaluated models is composed of 32 individuals. By default, the initial population is randomly generated. At each iteration, the number of created offspring λ is 64. The maximum number of evaluations is fixed to 10000. Each parent is selected by binary tournament selection. The recombination operator is applied with a probability of 0.60. Bit-flip mutation is used. Fitness proportional selection is used to build up the next population from the set of (μ+λ) individuals. These parameters (population size, stop criterion, probabilities, etc.) are chosen after an examination of some values previously used with success [1], [16].

We conduct our study on *NK* instances with *N*=96 bits varying the epistatic relations from *K*=0 to *K*=48 in increments of 4. We use landscapes with adjacent epistatic patterns among genes. For each combination of *N* and *K* we generated 30 random problem instances. Each algorithm was tested with each of these instances.
The algorithms are implemented inside MALLBA [2], a C++ software library fostering rapid prototyping of hybrid and parallel algorithms. Our computing system is a cluster of 8 machines with AMD Phenom8450 Triple-core Processor at 2GHz with 2 GB of RAM, linked by Gigabit, under Linux with 2.6.27-4GB kernel version.

VI. Experimental Analysis

Let us now proceed with the presentation of the results. The quality of a solution is measured by the percentage gap, i.e., the relative distance to the best solution obtained by GA_Adap (*best_sol*$_{GA\_Adap}$) and the best solution of each of the other algorithms, as described in Equation 2.

$$gap_{best} = \frac{best\_sol_{GA\_Adap} - GA_i}{best\_sol_{GA\_Adap}} \times 100 \quad (2)$$

Table 1 presents the performance of each GA version introduced in previous Section on each problem. Figures in the first two columns stand for the percentage gap regarding best solutions (column gap_{best}) and in the last two columns stand for the gap taking into account mean best solutions (column gap_{mean}). Positive values indicate superiority of the results obtained by the proposed self-adaptive GA, while negative values indicate the opposite situation.

From Table 1, it can be seen that the GA_UX's gap_{best} is positive for all the instances, from low to high levels of epistasis, which indicates that the solution qualities of GA_Adap are higher when compared with GA_UX. From the analysis of the gap_{best}'s values for GA_1X, there is not a clear winner; however in 7 of the 12

Table 1: gap values for traditional GAs

K	gap_{best}		gap_{mean}	
	GA_1X	**GA_UX**	**GA_1X**	**GA_UX**
0	-0.87	-0.17	0.35	0.53
4	-0.13	1.97	1.79	5.29
8	-6.28	1.59	1.55	4.03
12	0.66	0.66	2.17	3.39
16	0.67	0.65	3.61	6.35
20	0.65	0.64	2.73	4.38
24	0.65	0.67	3.35	4.32
28	0.65	0.64	2.33	3.37
32	-0.64	3.38	1.38	3.32
36	2.18	3.80	2.39	3.12
40	-0.76	6.49	0.85	1.51
44	-1.19	4.08	0.46	1.20
48	0.98	0.63	2.30	1.76

instances the values are positive, for problems with low and high level of epitasis. In the case of the gap_{mean} values, it is important to note that all gap values for GA_SX and GA_UX are positive, which means that in average the GA_Adap obtains a pool of best solutions with higher quality than traditional GAs. The previous observations stand for the superiority of our proposed adaptive algorithm, which using different recombination operators along the search contributes positively to the search process in the landscape. Another issue to emphasize is that the adaptive approach implicitly helps in minimizing the work involved in the selection of the appropriate recombination operator to solve the problem.

Figure 2 shows the mean number of generations to reach the best value for the GA_Adapt and traditional GAs. We can see that GA_Adapt presents a considerable higher numerical effort than traditional GAs, in order to find their best values. This observation is corroborated by statistical tests (ANOVA), indicating the significant differences among the algorithms in all instances (the respective *p*-values for ANOVA test are higher than $\alpha = 0.1$, the significance value).

One of the problems that remained to address is to determine whether any performance improvements were due to the adaptive mechanism, or due to the fact that their adaptive GA simply used more crossover operators. Consequently, in the following we concentrate in the adaptive mechanism itself to know if the adaptive behaviour was responsible for the performance improvement. A further analysis was done in order to address this issue. Instead of using a traditional GA, a reasonable study was to also run the GA with both one-point and uniform crossover. In this case, the algorithm, denoted as GA_Rand, randomly choices the recombination operator to apply each generation, using a 50/50 coin flips. By comparing the performance of this GA to the self-adaptive GA, the effect that the adaptive mechanism had on

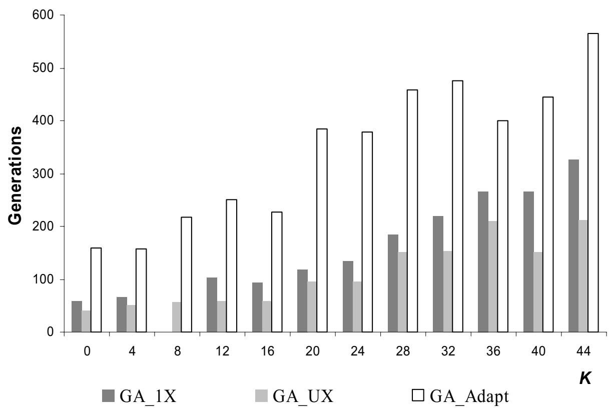


Figure 2: Mean number of generations to reach the best value for each *K* value.

Table 2: gap values for GA_Rand

K	0	4	8	12	16	20	24	28	32	36	40	44	48
gap_{best}	-2.1	1.00	2.35	0.66	0.66	0.65	0.68	0.65	2.18	0.64	2.48	-1.1	0.77
gap_{mean}	0.30	1.79	3.39	2.17	5.16	2.62	4.01	2.37	2.58	2.19	0.78	0.46	0.78

performance can be determined. Table 2 shows the gap values of this comparison. It can be seen that the values are positive, except for instance with *K*=44.

VII. Conclusions and Future Work

In this paper, we have presented an adaptive evolutionary algorithm to solve the NK landscapes, which selects the recombination operator to use during the solution of a problem. The operators considered in this work are the one-point and uniform crossover, which presents different levels of disruption of genetic material. The criterion to select which operator to apply is based on the use of a simple measure, such as the mean population fitness which gives some insides of the evolution process.

The results in this paper indicate, in general, that the adaptive algorithm appears to generate good performance results compared with traditional GAs. Analizing the GA_Adap performance against an algorithm randomly selecting between two crossover operators, we can see that our proposal was able to find better quality of solutions for the set of instances, no needing the traditional and costly ad-hoc pre-tuning of recombination operator.

As a future work, we propose to study more sophisticated criteria for the adaptation, also possibly including in this process the probability of recombination.

Furthermore, another extension to this work can be the analysis of the effectiveness of this adaptive approach over other complex problems.

Acknowledgments

We acknowledge the Universidad Nacional de La Pampa and the ANPCYT (Argentina) from which the authors receive continuous support.

References

[1] H. Aguirre and K. Tanaka. *Genetic algorithms on NK-landscapes: Effects of selection, drift, mutation, and recombination.* EvoWorkshops 2003, LNCS 2611, pp. 131-142, 2003.

[2] E. Alba et al. MALLBA: A Library of Skeletons for Combinatorial Optimisation, volume 2400 of LNCS, pages 927–932, 2002.

[3] Z. Bingul. *Adaptive genetic algorithms applied to dynamic multiobjective problems*. Applied Soft Computing, vol. 7, pp. 791–9, 2007.

[4] L. B. Booker, *Recombination Distributions for Genetic Algorithms*. Proc. of the Second Foundations of Genetic Algorithms Workshop, pp. 29-44, 1992.

[5] D. CH, Z. YF, C. WR. *Adaptive probabilities of crossover and mutation in genetic algorithms based on cloud model*. Proc. of 2006 IEEE Information Theory Workshop, pp. 710–713, 2006.

[6] K. De Jong, M.Potter, and W.M. Spears. *Using problem generators to explore the effects of epistasis*. Proc. of the Seventh International Conference on Genetic Algorithms (ICGA97), 1997.

[7] S.K. De Jong and W.M. Spears, *On the virtues of parametrized uniform crossover*. Proc. of Fourth International Conference on Genetic Algorithms, pp. 230–236, 1991.

[8] A.E. Eiben, R. Hinterding and Z. Michalewicz. *Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation,* vol. 3, pp. 124–141, 1999.

[9] L. Eshelman, R. Caruana, and J. Schaffer. *Biases in the crossover landscape*. Proc. of the Third International Conference on Genetic Algorithms, pp. 10–19, 1989.

[10] S.C. Ghosh, B.P. Sinha and N. Das. *Channel assignment using genetic algorithm based on geometric symmetry*. IEEE Transactions on Vehicular Technology, vol 52, pp. 860–75, 2003.

[11] J. Grefenstette. *A User's guide to GENESIS*. Navy Center for Applied Research in Artificial Intelligence, 1987.

[12] R.B. Heckendorn, S.Rana, and D.L. Whitley. *Test function generators as embedded landscapes*. Foundations of Genetic Algorithms 5, pp. 183-198, 1999.

[13] K. Frenken. *A fitness landscape approach to technological complexity, modularity, and vertical disintegration.* Structural Change and Economic Dynamics, vol. 17, pp. 288-305, 2006.

[14] S. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[15] S. A. Kauffman and S. Levin. *Towards a general theory of adaptive walks on rugged landscapes.* Journal of Theoretical Biology, vol. 128, pp. 11-45, 1987.

[16] C. Macken and A. Perelson. *Protein evolution on rugged landscapes*. National Academic

Science USA, vol. 86, pp. 6191-6195, 1989.
[17] D. Levinthal. *Adaptation on rugged landscapes*. Management Science, vol. 43, pp. 934-950, 1997.
[18] J.D. Schaffer, and A. Morishima. *An Adaptive Crossover Distribution Mechanism for Genetic Algorithms*. Proc. of the Second International Conference on Genetic Algorithms, pp. 36-40, 1987.
[19] R.E. Smith and J.E. Smith. *New methods for tunable, random landscapes.* Foundations of Genetic Algorithms 6. Morgan Kaufmann, 2001.
[20] W.M. Spears, and K.A. De Jong, *On the Virtues of Parameterized Uniform Crossover*. Proc. of the Fourth International Conference on Genetic Algorithms, pp. 230-236, 1991.
[21] W.M. Spears. *Adapting Crossover in Evolutionary Algorithms*. Proc of the Fourth Annual Conference on Evolutionary Programming, pp. 367-384, 1995.
[22] A. Stoltzfus. *Mutation-biased adaptation in a protein nk model*. Molecular Biology and Evolution, vol. 23(10), pp. 1852-1862, 2006.
[23] M. Srinivas, L.M. Patnaik. *Adaptive probabilities of crossover and mutation in genetic algorithms*. IEEE Trans. On Systems, Man and Cybernetics vol. 24, pp. 656-667, 1994.
[24] W. Syswerda, *Uniform Crossover in Genetic Algorithms*. Proc. of the Third International Conference on Genetic Algorithms, pp. 2-8, 1989.
[25] J.Welch and D. Waxman*.* *The NK model and population genetics*. Journal of Theoretical Biology, vol. 234(3), pp. 329-340, 2005.
[26] S. Yang, *Adaptive Crossover in Genetic Algorithms Using Statistics Mechanism*, Artificial Life VIII, pp. 182-185, 2002.